HW8

Tianyang Chen
N1307278
tc2820@nyu.edu

Lab 4: ICMP Pinger Lab

Code:

```
from socket import *
import os
import sys
import struct
import time
import select
import binascii

ICMP_ECHO_REQUEST = 8


def checksum(string):
    csum = 0
    countTo = (len(string) // 2) * 2
    count = 0

    while count < countTo:
        thisVal = ord(string[count + 1]) * 256 + ord(string[count])
        csum = csum + thisVal
        csum = csum & 0xffffffff
        count = count + 2

    if countTo < len(string):
        csum = csum + ord(string[len(string) - 1])
        csum = csum & 0xffffffff

    csum = (csum >> 16) + (csum & 0xffff)
    csum = csum + (csum >> 16)
    answer = ~csum
    answer = answer & 0xffff
    answer = answer >> 8 | (answer << 8 & 0xff00)
    return answer


def receiveOnePing(mySocket, ID, timeout, destAddr):
    timeLeft = timeout

    while 1:
        startedSelect = time.time()
        whatReady = select.select([mySocket], [], [], timeLeft)
        howLongInSelect = (time.time() - startedSelect)

        if whatReady[0] == []:    # Timeout
            return "Request timed out."
```

```
        timeReceived = time.time()
        recPacket, addr = mySocket.recvfrom(1024)

        icmpHeader = recPacket[20:28]
        icmpType, code, mychecksum, packetID, sequence = struct.unpack("bbHHh", icmpHeader)

        if type != 8 and packetID == ID:
            bytesInDouble = struct.calcsize("d")
            timeSent = struct.unpack("d", recPacket[28:28 + bytesInDouble])[0]
            return timeReceived - timeSent

        timeLeft = timeLeft - howLongInSelect

        if timeLeft <= 0:
            return "Request timed out."


def sendOnePing(mySocket, destAddr, ID):
    # Header is type (8), code (8), checksum (16), id (16), sequence (16)
    myChecksum = 0
    # Make a dummy header with a 0 checksum
    # struct -- Interpret strings as packed binary data
    header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, myChecksum, ID, 1)
    data = struct.pack("d", time.time())
    # Calculate the checksum on the data and the dummy header.
    myChecksum = checksum(str(header + data))
    # Get the right checksum, and put in the header
    if sys.platform == 'darwin':
        # Convert 16-bit integers from host to network byte order
        myChecksum = htons(myChecksum) & 0xffff
    else:
        myChecksum = htons(myChecksum)

    header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, myChecksum, ID, 1)
    packet = header + data
    mySocket.sendto(packet, (destAddr, 1))    # AF_INET address must be tuple, not str
    # Both LISTS and TUPLES consist of a number of objects
    # which can be referenced by their position number within the object.


def doOnePing(destAddr, timeout):
    icmp = getprotobyname("icmp")
    # SOCK_RAW is a powerful socket type. For more details: http://sockraw.org/papers/sock_raw
    mySocket = socket(AF_INET, SOCK_RAW, icmp)
    myID = os.getpid() & 0xFFFF    # Return the current process i
    sendOnePing(mySocket, destAddr, myID)
    delay = receiveOnePing(mySocket, myID, timeout, destAddr)
    mySocket.close()
    return delay
```

```
def ping(host, timeout=1):
    # timeout=1 means: If one second goes by without a reply from the server,
    # the client assumes that either the client's ping or the server's pong is lost
    dest = gethostbyname(host)
    print("Pinging " + dest + " using Python:")
    print("")
    # Send ping requests to a server separated by approximately one second
    while 1:
        delay = doOnePing(dest, timeout)
        print(delay)
        time.sleep(1)   # one second
    return delay

# ping("127.0.0.1")
# ping("www.google.com")
# ping("www.poly.edu")
# ping("www.baidu.com")
```

Results:

HW2 [C:\Users\acer\Downloads\computer networking\HW2] - C:\Users\acer\Downloads\computer networking\HW9\ICMP Pinger.py [...

File Edit View Navigate Code Refactor Run Tools VCS Window Help

HW9 ICMP Pinger.py

ICMP Pinger

ICMP Pinger.py

```python
        # SOCK_RAW is a powerful socket type. For more details: http://sockraw.org/papers/sock_raw
        mySocket = socket(AF_INET, SOCK_RAW, icmp)
        myID = os.getpid() & 0xFFFF  # Return the current process i
        sendOnePing(mySocket, destAddr, myID)
        delay = receiveOnePing(mySocket, myID, timeout, destAddr)
        mySocket.close()
        return delay


def ping(host, timeout=1):
    # timeout=1 means: If one second goes by without a reply from the server,
    # the client assumes that either the client's ping or the server's pong is lost
    dest = gethostbyname(host)
    print("Pinging " + dest + " using Python:")
    print("")
    # Send ping requests to a server separated by approximately one second
    while 1:
        delay = doOnePing(dest, timeout)
        print(delay)
        time.sleep(1)  # one second
    return delay

ping("127.0.0.1")
#ping("www.google.com")
#ping("www.poly.edu")
#ping("www.baidu.com")
```

Run: ICMP Pinger

```
"C:\Users\acer\Downloads\computer networking\HW9\venv\Scripts\python.exe" "C:/Users/acer/Downloads/computer networking/HW9/ICMP Pinger.py"
Pinging 127.0.0.1 using Python:

0.0
0.00100016593933
0.00100016593933
0.0
0.000999927520752
0.00100016593933
0.000999927520752
0.00100016593933

Process finished with exit code -1
```

4: Run 6: TODO Terminal Python Console Event Log

112:22 CRLF UTF-8

HW2 [C:\Users\acer\Downloads\computer networking\HW2] - C:\Users\acer\Downloads\computer networking\HW9\ICMP Pinger.py [...

File Edit View Navigate Code Refactor Run Tools VCS Window Help

HW9 ICMP Pinger.py

ICMP Pinger

ICMP Pinger.py

```python
        # SOCK_RAW is a powerful socket type. For more details: http://sockraw.org/papers/sock_raw
        mySocket = socket(AF_INET, SOCK_RAW, icmp)
        myID = os.getpid() & 0xFFFF  # Return the current process i
        sendOnePing(mySocket, destAddr, myID)
        delay = receiveOnePing(mySocket, myID, timeout, destAddr)
        mySocket.close()
        return delay


def ping(host, timeout=1):
    # timeout=1 means: If one second goes by without a reply from the server,
    # the client assumes that either the client's ping or the server's pong is lost
    dest = gethostbyname(host)
    print("Pinging " + dest + " using Python:")
    print("")
    # Send ping requests to a server separated by approximately one second
    while 1:
        delay = doOnePing(dest, timeout)
        print(delay)
        time.sleep(1)  # one second
    return delay

#ping("127.0.0.1")
ping("www.google.com")
#ping("www.poly.edu")
#ping("www.baidu.com")
```

Run: ICMP Pinger

```
"C:\Users\acer\Downloads\computer networking\HW9\venv\Scripts\python.exe" "C:/Users/acer/Downloads/computer networking/HW9/ICMP Pinger.py"
Pinging 172.217.164.132 using Python:

0.0199999809265
0.0269999504089
0.0130000114441
0.0160000324249
0.0190000534058
0.0169999599457
0.0269999504089
0.0159997940063

Process finished with exit code -1
```

4: Run 6: TODO Terminal Python Console Event Log

111:1 CRLF UTF-8

HW2 [C:\Users\acer\Downloads\computer networking\HW2] - C:\Users\acer\Downloads\computer networking\HW9\ICMP Pinger.py [...

File Edit View Navigate Code Refactor Run Tools VCS Window Help

HW9 ICMP Pinger.py

ICMP Pinger

ICMP Pinger.py

```python
        # SOCK_RAW is a powerful socket type. For more details: http://sockraw.org/papers/sock_raw
        mySocket = socket(AF_INET, SOCK_RAW, icmp)
        myID = os.getpid() & 0xFFFF  # Return the current process i
        sendOnePing(mySocket, destAddr, myID)
        delay = receiveOnePing(mySocket, myID, timeout, destAddr)
        mySocket.close()
        return delay


def ping(host, timeout=1):
    # timeout=1 means: If one second goes by without a reply from the server,
    # the client assumes that either the client's ping or the server's pong is lost
    dest = gethostbyname(host)
    print("Pinging " + dest + " using Python:")
    print("")
    # Send ping requests to a server separated by approximately one second
    while 1:
        delay = doOnePing(dest, timeout)
        print(delay)
        time.sleep(1)  # one second
    return delay

#ping("127.0.0.1")
#ping("www.google.com")
ping("www.poly.edu")
#ping("www.baidu.com")
```

Run: ICMP Pinger

```
"C:\Users\acer\Downloads\computer networking\HW9\venv\Scripts\python.exe" "C:/Users/acer/Downloads/computer networking/HW9/ICMP Pinger.py"
Pinging 35.172.89.115 using Python:

Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Process finished with exit code -1
```

4: Run 6: TODO Terminal Python Console Event Log

112:1 CRLF UTF-8

HW2 [C:\Users\acer\Downloads\computer networking\HW2] - C:\Users\acer\Downloads\computer networking\HW9\ICMP Pinger.py [...

File Edit View Navigate Code Refactor Run Tools VCS Window Help

HW9 ICMP Pinger.py

ICMP Pinger

ICMP Pinger.py

```python
        # SOCK_RAW is a powerful socket type. For more details: http://sockraw.org/papers/sock_raw
        mySocket = socket(AF_INET, SOCK_RAW, icmp)
        myID = os.getpid() & 0xFFFF  # Return the current process i
        sendOnePing(mySocket, destAddr, myID)
        delay = receiveOnePing(mySocket, myID, timeout, destAddr)
        mySocket.close()
        return delay


def ping(host, timeout=1):
    # timeout=1 means: If one second goes by without a reply from the server,
    # the client assumes that either the client's ping or the server's pong is lost
    dest = gethostbyname(host)
    print("Pinging " + dest + " using Python:")
    print("")
    # Send ping requests to a server separated by approximately one second
    while 1:
        delay = doOnePing(dest, timeout)
        print(delay)
        time.sleep(1)  # one second
    return delay

#ping("127.0.0.1")
#ping("www.google.com")
#ping("www.poly.edu")
ping("www.baidu.com")
```

Run: ICMP Pinger

```
"C:\Users\acer\Downloads\computer networking\HW9\venv\Scripts\python.exe" "C:/Users/acer/Downloads/computer networking/HW9/ICMP Pinger.py"
Pinging 104.193.88.77 using Python:

0.0950000286102
0.0929999351501
0.0950000286102
0.0940001010895
0.0939998626709
0.0889999866486
0.0980000495911
0.0889999866486

Process finished with exit code -1
```

4: Run 6: TODO Terminal Python Console Event Log

113:1 CRLF UTF-8

ICMP Traceroute Lab

Code:

```
from socket import *
import os
import sys
import struct
import time
import select
import binascii

ICMP_ECHO_REQUEST = 8
MAX_HOPS = 30
TIMEOUT = 2.0
TRIES = 2


# The packet that we shall send to each router along the path is the ICMP echo
# request packet, which is exactly what we had used in the ICMP ping exercise.
# We shall use the same packet that we built in the Ping exercise
def checksum(string):
    # In this function we make the checksum of our packet
    # hint: see icmpPing lab
    csum = 0
    countTo = (len(string) // 2) * 2
    count = 0

    while count < countTo:
        thisVal = ord(string[count + 1]) * 256 + ord(string[count])
        csum = csum + thisVal
        csum = csum & 0xffffffff
        count = count + 2

    if countTo < len(string):
        csum = csum + ord(string[len(string) - 1])
        csum = csum & 0xffffffff

    csum = (csum >> 16) + (csum & 0xffff)
    csum = csum + (csum >> 16)
    answer = ~csum
    answer = answer & 0xffff
    answer = answer >> 8 | (answer << 8 & 0xff00)
    return answer


def build_packet():
    # Header is type (8), code (8), checksum (16), id (16), sequence (16)
    myChecksum = 0
```

```
    myID = os.getpid() & 0xFFFF    # Return the current process i
    # Make a dummy header with a 0 checksum
    # struct -- Interpret strings as packed binary data
    header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, myChecksum, myID, 1)
    data = struct.pack("d", time.time())
    # Calculate the checksum on the data and the dummy header.
    myChecksum = checksum(str(header + data))
    # Get the right checksum, and put in the header
    if sys.platform == 'darwin':
        # Convert 16-bit integers from host to network byte order
        myChecksum = htons(myChecksum) & 0xffff
    else:
        myChecksum = htons(myChecksum)

    header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, myChecksum, myID, 1)
    packet = header + data
    return packet


def get_route(hostname):
    timeLeft = TIMEOUT
    for ttl in range(1, MAX_HOPS):
        for tries in range(TRIES):
            destAddr = gethostbyname(hostname)
            # Fill in start
            # Make a raw socket named mySocket
            icmp = getprotobyname("icmp")
            # SOCK_RAW is a powerful socket type. For more details: http://sockraw.org/papers/sock_raw
            mySocket = socket(AF_INET, SOCK_RAW, icmp)
            # Fill in end
            mySocket.setsockopt(IPPROTO_IP, IP_TTL, struct.pack('I', ttl))
            mySocket.settimeout(TIMEOUT)
            try:
                d = build_packet()
                mySocket.sendto(d, (destAddr, 0))
                t = time.time()
                startedSelect = time.time()
                whatReady = select.select([mySocket], [], [], timeLeft)
                howLongInSelect = (time.time() - startedSelect)

                if whatReady[0] == []:   # Timeout
                    print("* * * Request timed out.")

                recvPacket, addr = mySocket.recvfrom(1024)
                timeReceived = time.time()
                timeLeft = timeLeft - howLongInSelect

                if timeLeft <= 0:
                    print("* * * Request timed out.")
```

```
        except timeout:
            continue

        else:
            # Fill in start
            # Fetch the icmp type from the IP packet
            icmpHeader = recvPacket[20:28]
            types, code, mychecksum, packetID, sequence = struct.unpack("bbHHh", icmpHeader)
            # Fill in end

            if types == 11:
                # bytes = struct.calcsize("d")
                # timeSent = struct.unpack("d", recvPacket[28:28 + bytes])[0]
                print(" %d rtt=%.0f ms %s" % (ttl, (timeReceived - t) * 1000, addr[0]))

            elif types == 3:
                # bytes = struct.calcsize("d")
                # timeSent = struct.unpack("d", recvPacket[28:28 + bytes])[0]
                print(" %d rtt=%.0f ms %s" % (ttl, (timeReceived - t) * 1000, addr[0]))

            elif types == 0:
                bytes = struct.calcsize("d")
                timeSent = struct.unpack("d", recvPacket[28:28 + bytes])[0]
                print(" %d rtt=%.0f ms %s" % (ttl, (timeReceived - timeSent) * 1000, addr[0]))
                return

            else:
                print("error")
            break

        finally:
            mySocket.close()

# get_route("127.0.0.1")
# get_route("www.google.com")
# get_route("www.poly.edu")
# get_route("www.baidu.com")
```

Results:

HW2 [C:\Users\acer\Downloads\computer networking\HW2] - C:\Users\acer\Downloads\computer networking\HW9\ICMP Traceroute....

File Edit View Navigate Code Refactor Run Tools VCS Window Help

HW9 ICMP Traceroute.py ICMP Traceroute

ICMP Pinger.py ICMP Traceroute.py

```
            # timeSent = struct.unpack("d", recvPacket[28:28 + bytes])[0]
            print(" %d rtt=%.0f ms %s" % (ttl, (timeReceived - t) * 1000, addr[0]))

        elif types == 3:
            # bytes = struct.calcsize("d")
            # timeSent = struct.unpack("d", recvPacket[28:28 + bytes])[0]
            print(" %d rtt=%.0f ms %s" % (ttl, (timeReceived - t) * 1000, addr[0]))

        elif types == 0:
            bytes = struct.calcsize("d")
            timeSent = struct.unpack("d", recvPacket[28:28 + bytes])[0]
            print(" %d rtt=%.0f ms %s" % (ttl, (timeReceived - timeSent) * 1000, addr[0]))
            return

        else:
            print("error")
        break

    finally:
        mySocket.close()


get_route("127.0.0.1")
#get_route("www.google.com")
#get_route("www.poly.edu")
#get_route("www.baidu.com")
```

Run: ICMP Traceroute

```
"C:\Users\acer\Downloads\computer networking\HW9\venv\Scripts\python.exe" "C:/Users/acer/Downloads/computer networking/HW9/ICMP Traceroute.py"
 1 rtt=0 ms 127.0.0.1

Process finished with exit code 0
```

4: Run 6: TODO Terminal Python Console

128:1 CRLF UTF-8

HW2 [C:\Users\acer\Downloads\computer networking\HW2] - C:\Users\acer\Downloads\computer networking\HW9\ICMP Traceroute....

File Edit View Navigate Code Refactor Run Tools VCS Window Help

HW9 ICMP Traceroute.py

ICMP Traceroute

ICMP Pinger.py | ICMP Traceroute.py

```python
            elif types == 3:
                # bytes = struct.calcsize("d")
                # timeSent = struct.unpack("d", recvPacket[28:28 + bytes])[0]
                print(" %d rtt=%.0f ms %s" % (ttl, (timeReceived - t) * 1000, addr[0]))

            elif types == 0:
                bytes = struct.calcsize("d")
                timeSent = struct.unpack("d", recvPacket[28:28 + bytes])[0]
                print(" %d rtt=%.0f ms %s" % (ttl, (timeReceived - timeSent) * 1000, addr[0]))
                return

            else:
                print("error")
            break

        finally:
            mySocket.close()


#get_route("127.0.0.1")
get_route("www.google.com")
#get_route("www.poly.edu")
#get_route("www.baidu.com")
```

Run: ICMP Traceroute

```
"C:\Users\acer\Downloads\computer networking\HW9\venv\Scripts\python.exe" "C:/Users/acer/Downloads/computer networking/HW9/ICMP Traceroute.py"
 1 rtt=4 ms 10.0.0.1
 2 rtt=15 ms 96.120.11.49
 3 rtt=16 ms 68.85.80.249
 4 rtt=12 ms 96.110.6.101
 5 rtt=15 ms 96.110.6.5
 6 rtt=12 ms 162.151.60.233
 7 rtt=15 ms 68.86.90.57
 8 rtt=16 ms 68.86.82.214
 9 rtt=14 ms 50.208.232.110
 * * * Request timed out.
 * * * Request timed out.
 11 rtt=20 ms 216.239.54.124
 12 rtt=17 ms 108.170.240.112
 13 rtt=16 ms 216.239.50.93
 14 rtt=14 ms 216.239.50.96
 15 rtt=16 ms 108.170.246.33
 16 rtt=23 ms 216.239.49.47
 17 rtt=16 ms 172.217.8.4
```

4: Run 6: TODO Terminal Python Console

18:02

HW2 [C:\Users\acer\Downloads\computer networking\HW2] - C:\Users\acer\Downloads\computer networking\HW9\ICMP Traceroute....

File Edit View Navigate Code Refactor Run Tools VCS Window Help

HW9 ICMP Traceroute.py

ICMP Pinger.py | ICMP Traceroute.py

```python
                print(" %d rtt=%.0f ms %s" % (ttl, (timeReceived - t) * 1000, addr[0]))

            elif types == 0:
                bytes = struct.calcsize("d")
                timeSent = struct.unpack("d", recvPacket[28:28 + bytes])[0]
                print(" %d rtt=%.0f ms %s" % (ttl, (timeReceived - timeSent) * 1000, addr[0]))
                return

            else:
                print("error")
            break

        finally:
            mySocket.close()


# get_route("127.0.0.1")
# get_route("www.google.com")
get_route("www.poly.edu")
#get_route("www.baidu.com")
```

Run: ICMP Traceroute

```
"C:\Users\acer\Downloads\computer networking\HW9\venv\Scripts\python.exe" "C:/Users/acer/Downloads/computer networking/HW9/ICMP Traceroute.py"
 1 rtt=4 ms 10.0.0.1
 2 rtt=14 ms 96.120.11.49
 3 rtt=16 ms 68.85.80.249
 4 rtt=13 ms 96.110.6.101
 5 rtt=20 ms 96.110.6.5
 6 rtt=14 ms 162.151.60.233
 7 rtt=20 ms 68.86.90.57
 8 rtt=15 ms 96.110.32.173
 9 rtt=15 ms 96.110.32.190
 10 rtt=20 ms 66.208.233.18
 * * * Request timed out.
 * * * Request timed out.
 * * * Request timed out.
 * * * Request timed out.
 * * * Request timed out.
 * * * Request timed out.
 * * * Request timed out.
 * * * Request timed out.
 * * * Request timed out.
```

4: Run | 6: TODO | Terminal | Python Console

131:1 CRLF UTF-8

HW2 [C:\Users\acer\Downloads\computer networking\HW2] - C:\Users\acer\Downloads\computer networking\HW9\ICMP Traceroute....

File Edit View Navigate Code Refactor Run Tools VCS Window Help

HW9 ICMP Traceroute.py

ICMP Traceroute

ICMP Pinger.py | ICMP Traceroute.py

```python
                print(" %d rtt=%.0f ms %s" % (ttl, (timeReceived - t) * 1000, addr[0]))

            elif types == 0:
                bytes = struct.calcsize("d")
                timeSent = struct.unpack("d", recvPacket[28:28 + bytes])[0]
                print(" %d rtt=%.0f ms %s" % (ttl, (timeReceived - timeSent) * 1000, addr[0]))
                return

            else:
                print("error")
            break

        finally:
            mySocket.close()


# get_route("127.0.0.1")
# get_route("www.google.com")
# get_route("www.poly.edu")
get_route("www.baidu.com")
```

Run: ICMP Traceroute

```
"C:\Users\acer\Downloads\computer networking\HW9\venv\Scripts\python.exe" "C:/Users/acer/Downloads/computer networking/HW9/ICMP Traceroute.py"
 1 rtt=5 ms 10.0.0.1
 2 rtt=18 ms 96.120.11.49
 3 rtt=13 ms 68.85.80.249
 4 rtt=12 ms 96.110.6.101
 5 rtt=19 ms 96.110.6.5
 6 rtt=13 ms 162.151.60.233
 7 rtt=14 ms 68.86.90.57
 8 rtt=15 ms 96.110.32.173
 9 rtt=15 ms 96.110.32.190
 10 rtt=21 ms 66.208.233.50
 11 rtt=99 ms 216.6.87.111
 12 rtt=90 ms 66.110.59.1
 13 rtt=91 ms 66.110.59.9
 14 rtt=86 ms 63.243.251.2
 15 rtt=89 ms 209.58.86.30
 16 rtt=93 ms 104.193.88.29
 17 rtt=91 ms 104.193.88.77

Process finished with exit code 0
```

4: Run 6: TODO Terminal Python Console

18:04